Neural Networks, Vol. 6, pp. 177–187, 1993
Printed in the USA. All rights reserved.

0893-6080/93 $6.00 + .00

ORIGINAL CONTRIBUTION

The Error-Bounded Descriptional Complexity of Approximation Networks

RÜDIGER W. BRAUSE

J.W. Goethe-University

(*Received* 8 *January* 1992; *accepted* 10 *June* 1992)

Abstract—*It is well known that artificial neural nets can be used as approximators of any continuous functions to any desired degree and therefore be used, e.g., in high-speed, real-time process control. Nevertheless, for a given application and a given network architecture, the nontrivial task remains to determine the necessary number of neurons and the necessary accuracy (number of bits) per weight for a satisfactory operation which are critical issues in VLSI and computer implementations of nontrivial tasks. In this paper the accuracy of the weights and the number of neurons are seen as general system parameters which determine the maximal approximation error by the absolute amount and the relative distribution of information contained in the network. We define as the "error-bounded network descriptional complexity" the minimal number of bits for a class of approximation networks which show a certain approximation error and achieve the conditions for this goal by the new principle of "optimal information distribution." For two examples, a simple linear approximation of a nonlinear, quadratic function and a nonlinear approximation of the inverse kinematic transformation used in robot manipulator control, the principle of optimal information distribution gives the optimal number of neurons and the resolutions of the variables, i.e., the minimal amount of storage for the neural net.*

Keywords—Kolmogorov complexity, ε-Entropy, Rate-distortion theory, Approximation networks, Information distribution, Weight resolutions, Kohonen mapping, Robot Control.

1. INTRODUCTION

One of the most common tasks of artificial neural nets is the approximation of a given function by the superposition of several functions of single neurons. This is especially useful for real-time, high-speed controller for industrial process control which are often implemented with descrete electronic components.

Similar to the well-known theorem of Stone-Weierstraß, Hornik, Stinchcomb, and White (1989) have shown (see also, e.g., Giroso & Poggio, 1990, for the property of "best approximation" function and regularization networks) that in a compact interval, every function can be arbitrarily closely approximated in the L_∞-Norm by a two layer neural network (see Figure 1) when a sufficiently large number m of units is provided and each unit output function $S(\cdot)$ satisfy the conditions $S(-\infty) = 0$, $S(\infty) = 1$.

Sufficiently large—What does this mean? How do we select the appropriate number of neuronal processors for a certain application and implementation? Let us consider only the case of a one-dimensional output approximation, as it was done in the paper of Hornik et al., 1993. Analogous results hold for multioutput networks, i.e., vector-valued functions.

1.1. Error-Bounded Descriptional Complexity

An important example for a feedforward network is an approximation network. Let us regard an approximation $\hat{f}$ of the function f: $RI^n \rightarrow RI$ in a compact interval $C_c IR^n$; not necessarily the best possibile approximation function. For example, this can be done by the two-layer neural network of Figure 1. Let the maximal absolute error of this approximation be δ_f with

$$\delta_f = \max_{\mathbf{x} \in C} |f(\mathbf{x}) - \hat{f}(\mathbf{x})| \tag{1.1}$$

for a given approximation function $\hat{f}$.

We can regard the approximation error as a kind of discretization error. Denoting the complete value range of f with

$$V_f = |f_{\max} - f_{\min}|, \quad f_{\max} = \max_{\mathbf{x} \in C} f(\mathbf{x}), \quad f_{\min} = \min_{\mathbf{x} \in C} f(\mathbf{x})$$

we can conclude that there are only V_f / d distinguishable, fixed states of the variable f which differ by an

Requests for reprints should be sent to Dr. R. Brause, J. W. Goethe-University, FB20, Postbox 11 19 32, D-6000 Frankfurt 11, Germany.

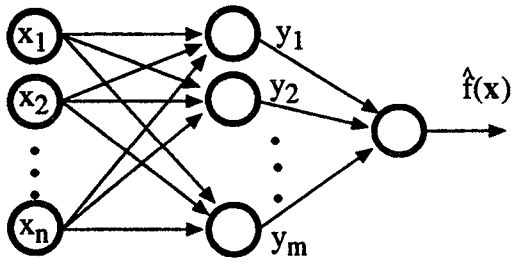


FIGURE 1. A two-layer universal approximation network.

increment of $d = 2\delta_f$. All other states are undistinguishable from deviations of the fixed states. Thus, since we do not know the input distribution of $\{\mathbf{x}\}$ and therefore not the error distribution, the output has minimal

$$I_{\text{out}} = \log_2(V/d) \tag{1.2}$$

bits of information.

In the neural network, the approximation $\hat{f}(\mathbf{x})$ depends also on the set w of all data bits (information) of the weights $\{\mathbf{w}\}$ of all neurons, denoted by $\hat{f}(\mathbf{x}, w)$. The system parameters which determine the error of the approximation are, on the one hand, the resolution of the weights or its information content

$$I_w = \log_2(V_w/d_w) \quad \text{with the weight increment } d_w \tag{1.3}$$

and on the other hand the number m of neurons.

Certainly, when we increase the number of neurons and the number of bits per neuron, the approximation will become better and the error will decrease. Nevertheless, for a certain system with a finite amount of information storage capacity (such as a digital computer) the network description information (system state) will be limited. For constant system information neither one neuron with high-resolution weights nor many neurons with one bit weights will give the optimal answer; the solution is in between the range, cf. Figure 6.

Therefore, we have to solve the problem: What is the best information distribution, i.e., what is the best choice for the parameters m and I_w to maximize the distribution, i.e., what is the best choice for the parameters m and I_w to maximize the Information I_{out} or to minimize the approximation error δ_f, using a fixed amount of system information I_{sys}?

If we regard the approximation network as a channel, we can formulate the whole problem as the task for the maximization of the *transinformation* between input and output, i.e., the determination of the channel capacity. This was done in Brause (1991). Now, let us take a different and interesting road to the solution of the problem.

The system information I_{sys} is just the number of bits we use for the representation of the weights of the neurons. Since the neural algorithm $\hat{f}$ (the network architecture) remains the same for different weight resolutions and different numbers of neurons, the minimization of the system information is identical to the minimization of the data size of the weights used in the network, apart from an additive constant. We can think of the neural network function $\hat{f}$ as a kind of interpreter or decoding function of the weights $\{\mathbf{w}\}$ on the condition that an input object $\mathbf{x}$ is given. The *descriptional complexity* (see, e.g., Li & Vitanyi, 1990) $K_{\hat{f}}(f|\mathbf{x})$ of the object f (the wanted output $f(\mathbf{x})$ of the approximation network) with respect to $\hat{f}$, conditional to $\mathbf{x}$, can be defined by

$$K_{\hat{f}}(f|\mathbf{x}) = \min\{|w| : w \in \{0,1\}^* \text{ and } \hat{f}(w, \mathbf{x}) = f(\mathbf{x})\}$$

descriptional complexity (1.4)

and $K_{\hat{f}}(f|\mathbf{x})$ becomes infinity if there are no such w. The set of weight bits w of the network containing overall $|w|$ bits can be seen as the necessary information on which the output $\hat{f}(w, \mathbf{x})$ is based: Different information w will result in different approximations. In contrast to computer programs which produce binary strings f on the input of binary strings $\mathbf{x}$, the neural program is not able to approximate the wanted output $f(\mathbf{x})$ always exactly—generally, there is a finite error depending on the number of bits for the weights used. Thus, we can define the *error-bounded descriptional complexity* $K_{\hat{f},\varepsilon}$ by

$$K_{\hat{f},\varepsilon}(f|\mathbf{x}) = \min\{|w| : w \in \{0,1\}^* \text{ and } |\hat{f}(w, \mathbf{x}) - f(\mathbf{x})| < \varepsilon\} \tag{1.5}$$

where the minimum is taken again over the sum of the number of bits of all the weights in the network at all possible assignment of bits to the weights. For the whole interval, the number

$$|w| = \max_{\mathbf{x} \in C} K_{\hat{f},\varepsilon}(f|\mathbf{x}) \tag{1.6}$$

is the minimal number of bits in the network necessary to guarantee a maximal approximation error of $\delta < \varepsilon$ for the whole input interval. Our main task of computing the descriptional complexity for a concrete neural network consists of computing just this number: The minimal amount of information to describe the state of the network.

The basic idea behind this is not new. The problem of encoding an information source with the minimal number of bits without exceeding a certain error or fidelity criterion was first introduced by Shannon and Weaver (1949) and is known as the *rate-distortion problem* (see, e.g., Gallager, 1968).

Let us now consider another connection to a neighbour research field. Each number of bits for the weights in the network architecture $\hat{f}$ results in a different approximation function $\hat{f}(w)$. For a fixed number $|w|$ of bits only a fixed number of functions $\hat{f}(w)$ exists. This number is the number of possible "neural programs" and, for a certain distribution of the bits to the weights, is equal to the number of possible states of the

set w of all bits. If we further restrict the class $\{\hat{f}(w)\}$ by a certain error constraint, the logarithm to base 2 of the number N_f of such functions is the number $|w|$ of bits:

$$H_{\hat{f}\cdot,\varepsilon} = \log_2 N_f. \tag{1.7}$$

Therefore, our problem of error-constraint minimization of I_{sys} becomes the problem of the minimization of the number of elements in the ε-cover of the functional class. The logarithm to base 2 of this number was termed "ε-Entropy" by Kolmogorov and Tihomirov, 1959. There is not much literature on the subject of neural networks. For binary networks, Williamson (1991) computed some lower and upper limits of the ε-Entropy; the determination of the ε-Entropy for a feedforward neural network is still missing.

In this paper, we do not only determine I_{sys}, the minimal number of bits for a given maximal approximation error, for a fixed assignment of bits to weights as it is necessary to determine the ε-Entropy, but we also change the assignment in order to minimize further the approximation error by the means of the principle of optimal information distribution.

2. OPTIMAL INFORMATION DISTRIBUTION

As we know, the task of computing the error-bounded description complexity for approximation networks, i.e., the system information I_{sys} when a certain error is fixed, is equivalent to the task of computing the minimal error when a certain I_{sys} is given. When we have different weights, w_i, W_j, t_i, T, with different resolutions (number of bits per weight) in the network, the question becomes: What is the best choice for the parameters m and I_w to minimize the approximation error δ_f, using a fixed amount of system information I_{sys}?

The solution to this question is provided by the approach of an optimal information distribution of the neural network parameters. For this purpose let us denote the parameters m, I_w, ... as general system parameters $c_1, \ldots, c_k$.

2.1. The Principle of Optimal Information Distribution

Let us first derive the conditions for the optimal system parameters by some plausible considerations, presented in Brause (1989), which should give a feeling for the subject and insight into the mechanism involved. The rigorous, conventional mathematical approach will be covered in Section 2.2.

Assume on the one hand that we transfer a fixed, small amount of information from one parameter to another (e.g., more neurons and less bits per weight) and we will find the approximation error decreased. In this case the information distribution induced by the parameter values of $c_1, \ldots, c_k$ was not optimal; the new one is better. Let us assume that we find that the error δ_f has increased, then the information distribution is not optimal; by making the inverse transfer we can also decrease δ_f. All subsequent changes in a nonoptimal information distribution will further reduce the error until we reach a minimum. Thus, in a restricted system we have at least one local minimum of error. This extremum can be characterized by the following principle: In an *optimal information distribution* a small (virtual) change in the distribution (a change in $c_1, \ldots, c_k$) neither increases nor decreases the performance error δ_f.

A small increment of additional information ΔI_{sys} in the system will produce a change $\Delta\delta_f$ in the maximal output error

$$\Delta\delta_f = \Delta I_{\text{sys}} \frac{\partial}{\partial I_{\text{sys}}} \delta_f = \Delta I_{\text{sys}} \sum_{i=1}^{k} \frac{\partial}{\partial c_i} \delta_f(c_1, \ldots, c_k) \frac{\partial c_i}{\partial I_{\text{sys}}}. \tag{2.1}$$

Each term in the sum of eqn (2.4) represents an information contribution of a system parameter when we increase the overall system information I_{sys}. According to the principle above, an optimal distribution is given when all terms in the sum, i.e., all information contributions of the system parameters are equal.

$$\frac{\partial\delta_f}{\partial c_1}\frac{\partial c_1}{\partial I_{\text{sys}}} = \ldots = \frac{\partial\delta_f}{\partial c_k}\frac{\partial c_k}{\partial I_{\text{sys}}}. \tag{2.2}$$

The k independent terms gives us $(k-1)$ equations for k variables $c_1, \ldots, c_k$, leaving us with a degree of freedom of one. So, choosing the amount of available information storage $I_{\text{sys}}(c_1, \ldots, c_k) = I_0$, the parameters $c_1, \ldots, c_k$ are fixed and the smallest error δ_f for the particular application will result. On the other hand, for a certain maximal error, a certain amount of network information is necessary.

2.2. Optimal System Parameters

Now we want to compare the principle above to a more conventional mathematical approach.

The maximal error δ_f is a multivariate function $\delta_f(c_1, \ldots, c_k)$. We will look for the minimal error of the system using only a certain amount of system information and search an optimal parameter tuple $(c_1^*, \ldots, c_k^*)$ such that

$$\delta_f(c_1^*, \ldots, c_k^*) = \min_{c_1, \ldots, c_k} \delta_f(c_1, \ldots, c_k) \tag{2.3}$$

which is accompanied by the restriction that the whole information I_{sys} in the system should not be changed during the maximization process

$$I_{\text{sys}}(c_1, \ldots, c_k) = I_0 = \text{const.} \tag{2.4}$$

By these two conditions, the relative minimum (2.3) of the multivariate function δ_f is searched under the restriction of eqn (2.4). The standard method to get the local extrema of a constrained function is the

method of Lagrange multipliers. For this purpose let us define the differentiable functions:

$$L(c_1, \ldots, c_k, \lambda) \equiv \delta_f(c_1, \ldots, c_k) + \lambda I(c_1, \ldots, c_k)$$
$$\text{with} \quad I(c_1, \ldots, c_k) \equiv I_{sys}(c_1, \ldots, c_k) - I_0 = 0. \quad (2.5)$$

Since the function L includes the restrictions, the necessary conditions for a relative extremum of the function gives us the necessary conditions for optimal values of the system parameters

$$\frac{\partial}{\partial c_1} L(c_1^*) = 0, \ldots, \quad \frac{\partial}{\partial c_k} L(c_k^*) = 0,$$
$$\frac{\partial}{\partial \lambda} L(\lambda^*) = 0. \quad (2.6)$$

The conditions above transform to the equations

$$\frac{\partial}{\partial c_1} \delta_f(c_1^*) + \lambda \frac{\partial}{\partial c_1} I(c_1^*) = 0, \ldots,$$
$$\frac{\partial}{\partial c_k} \delta_f(c_k^*) + \lambda \frac{\partial}{\partial c_k} I(c_k^*) = 0 \quad (2.7a)$$
$$I(c_1^*, \ldots, c_k^*) = 0. \quad (2.7b)$$

Let us assume that the function $I(c_1, \ldots, c_k)$ is invertible for each system parameter. Then we know that

$$\frac{\partial}{\partial c_i} I(c_i) = \frac{\partial}{\partial c_i} I_{sys}(c_i) = \left[\frac{\partial c_i}{\partial I_{sys}(c_i)}\right]^{-1} \quad (2.8)$$

and the conditions (2.7a) become

$$\frac{\partial}{\partial c_1} \delta_f(c_1^*) \frac{\partial c_1}{\partial I_{sys}} = -\lambda = \ldots = \frac{\partial}{\partial c_k} \delta_f(c_k^*) \frac{\partial c_k}{\partial I_{sys}} \quad (2.9a)$$
$$I_{sys}(c_1^*, \ldots, c_k^*) = I_0. \quad (2.9b)$$

Equation (2.9a) shows that for the conditions of an optimal information distribution, all the terms in eqn (2.9a) should be equal: This is the *principle of optimal information distribution* as it is stated above in Section 2.1 and expressed in eqn (2.2). The last condition (2.9b) is just our well-known restriction (2.4).

It is well known that the mechanism of Lagrangian multipliers does not provide a general solution, what kind of extremum we have; the decision of whether $\mathbf{c}^*$ is a relative maximum, minimum, or a saddle point must be decided according to the application problem. In our case, the decision is clear: According to Section (2.1) there exists at least one local minimum. Since we have only one extremum in every application example in Section 3, these extrema must be minima.

3. APPLICATION EXAMPLES

In this section we first want to demonstrate the procedure above by a very simple example: The approximation of a quadratic form by a polyline or linear splines. Throughout this example, all design decisions (choice of value ranges, etc.) are taken for demonstration purposes only; the whole example is simple enough to be verified analytically by the interested reader.

The section afterwards is intended to be more realistic but is also more complicated: Here we show the use of the information distribution principle for the application example of a robot control algorithm which uses nonlinear, learned mapping. Since the computations are quite complex, they are given only as an overview. The more interested reader is referred to Brause (1989). Let us now regard the simplified example.

3.1. The Approximation of a Simple Nonlinear Function

Let us consider the simple nonlinear function $f(x) \equiv ax^2 + b$. The approximation of this function can be accomplished by a network with one input x shown in Figure 2. Another version of the quadratic function is the logistic function $x(t+1) = f(x(t)) \equiv ax(t)(1 - x(t)) = ax(t) - ax(t)^2$, which yields deterministic chaotic behaviour in the interval [0, 1] for some values of the parameter a, see, e.g., Baker and Gollub (1990). This system can be approximated by the network of Figure 2, using an additional, direct input $W_0 x$ for the second layer to model the linear term ax of the logistic function. The learning of the weights and thresholds by the Backpropagation Algorithm was demonstrated by Lapedes and Farber (1987).

Let us return to our example of the quadratic func-

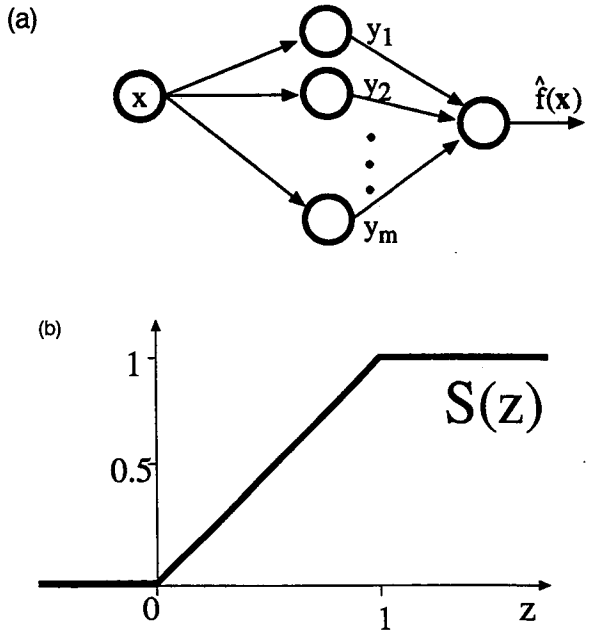


FIGURE 2. (a) The network for approximating the function $f(x) = ax^2 + b$; (b) the limited linear neural unit output function $S(z)$. As you can see, it satisfies the conditions $S(-\infty) = 0$ and $S(\infty) = 1$.

tion $f(x) = ax^2 + b$. Each neuron of the network of Figure 2 has the *output function* $y_i = S(z_i)$ with the *activation function* (potential) z_i

$$z_i = \sum_j w_{ij}x_j \tag{3.1}$$

which becomes for the first layer

$$z_i = w_i x + t_i \quad \text{with the threshold } t_i \tag{3.2}$$

and for the second layer

$$\hat{f}(x) = \hat{f}(x, \mathbf{W}, T, \mathbf{w}, \mathbf{t}) = \sum_i W_i S(z_i) + T \tag{3.3}$$

with the weight tuples $\mathbf{w} = (w_1, \ldots, w_m)$ and thresholds $\mathbf{t} = (t_1, \ldots, t_m)$ for the first layer and $\mathbf{W} = (W_1, \ldots, W_m)$ and threshold T for the second layer. Let us assume that we use a simple limited linear output function as squashing function

$$S(z_i) = \begin{cases} 1 & 1 < z_i \\ z_i & 0 \le z_i \le 1 \\ 0 & z_i < 0. \end{cases} \tag{3.4}$$

The definition (3.4) satisfies the conditions $S(\infty) = 1$, $S(-\infty) = 0$ of Hornik et al. (1989) and is shown in Figure 2 (6). The choice of a linear output function is not only motivated by its analytical simplicity but also by the fact that it can be easily implemented by an ordinary linear electronic amplifier with output signal limits.

Let us assume that all weights have converged by a proper algorithm for an approximation of the nonlinear function by linear splines. If the linear interval $0 \leqslant z_i \leqslant 1$ of each neuron is identical to the others, the superposition will yield again only a line, resulting in a bad approximation of a parabola by one line. To obtain as many approximating lines as possible, the algorithm has to make all intervals different. Since the output of each neuron is only linear in x when $z_i \in [0, 1]$ and otherwise it is constant 0 or 1, it is a good choice for the approximation to divide the whole input interval $[X_0, X_1]$ by the m neurons of the first layer into m equal (see Appendix A) intervals $\Delta x = [x_i - \Delta x/2, x_i + \Delta x/2]$ with $x_i = X_0 + i\Delta x - \Delta x/2$. The segmented normalized variable $z_i \in [0, 1]$ is $\frac{1}{2}$ for each x_i at the middle of the interval i.

In the second layer, the output z_i becomes weighted by the weight W_i. Together with an offset of the previous intervals, it represents the linear part of the approximation function $\hat{f}(x)$ in the interval $[x_i - \Delta x/2, x_i + \Delta x/2]$:

$$\hat{f}(x) = \sum_{i=1}^{m} W_i S(z_i) + T = \sum_{i=1}^{k-1} W_i + T + W_k S(z_k). \tag{3.5}$$

offset *linear part*

The resulting approximation is shown in Figure 3. The

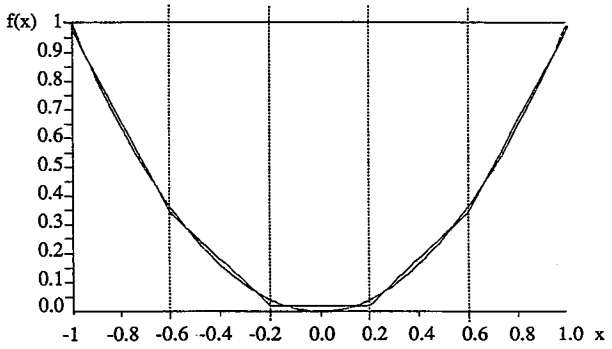


FIGURE 3. The nonlinear function $f(x)$ and its approximation by linear splines, the polygon of the superposition of the linear neuronal output. The dotted vertical lines denote the input interval borders for linear output of each neuron.

corresponding values for w_i, t_i, W_i, and T can be easily calculated (see Brause, 1991). From the conditions of eqn (3.4), we can conclude that the value of z_i at $x_i - \Delta x/2$ is zero and at $x_i + \Delta x/2$ it is one.

Therefore, by eqn (3.2) we get

$$w_i = 1/\Delta x = m/(X_1 - X_0) \tag{3.6a}$$

and

$$t_i = -w_i(x_i - \Delta x/2) = X_0/\Delta x + 1 - i$$
$$= -\mathrm{mx}_i/(X_1 - X_0) + \tfrac{1}{2}. \tag{3.6b}$$

Let us choose W_i such that in each segment the spline is parallel to the tangent of $f(x)$ in x_i

$$\left.\frac{\partial f(x)}{\partial x}\right|_{x_i} = \left.\frac{\partial(ax^2 + b)}{\partial x}\right|_{x_i} = 2ax_i = \Delta y/\Delta x.$$

Since the output $S(z)$ is normalized between 0 and 1, we have to choose the weights W_i as the normalized tangent $\Delta y/1$. Therefore, the weights become

$$W_i = \Delta y/1 = 2ax_i\Delta x. \tag{3.6c}$$

Then the basic threshold T becomes the offset of the approximation at X_0 (see Figure 3). Using eqn (A.1) we get

$$T = f(X_0) - \delta^{\mathrm{lin}} = aX_0^2 + b - a/2(\Delta x/2)^2. \tag{3.6d}$$

Example: For a net of $m := 5$ neurons we get for $a = 1$, $b = 0$, $X_0 = -1$, $X_1 = 1$ with $\Delta x = 0.4$ five nonoverlapping intervals $[-1, -.6]$, $[-.6, -.2]$, $[-.2, +.2]$, $[+.2, +.6]$, $[+.6, +1]$ with $x_i = \{-.8, -.4, 0, +.4, +.8\}$, $W_i = \{-.64, -.32, 0, +.32, +.64\}$, and $w_i = 2.5$, $t_i = \{+2.5, +1.5, +0.5, -0.5, -1.5\}$, $T = 0.98$. The maximal linear approximation error $\delta^{\mathrm{lin}} = 0.02$ has the same order as in the simulation results of Lapedes and Farber (1987).

In Figure 4, the superposition of the approximating function by the individual neural output $S_i(x)$ is shown. Each neuron has its linear output restricted to its input interval, otherwise it remains constant.

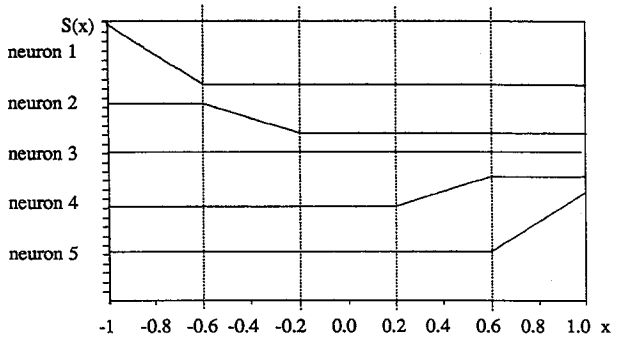


FIGURE 4. The individual neural approximation outputs for a = 1, b = 0, m = 5. By the dotted lines the input is divided in exclusive intervals, again. In each input interval, there is a neuron with a linear output. The weighted superposition of all outputs becomes a polygon. For the sake of clarity the output responses of the neurons are shifted vertically in the drawing.

Due to Figure 3 (and Figure A.1), we might suppose that the error of the approximation does not remain constant but has minimal and maximal values. This is confirmed in Figure 5 for the example of five neurons. In some control applications we are not interested in the mean error over the interval (which is approximately zero in the example above) but in the maximal error that can occur. Thus, we do not aim to minimize neither the average error nor the mean squared error of the approximation, but to minimize the *maximal* absolute error of eqn (1.1), i.e., the maximal squared error. As the error of the linear approximation we consider therefore the maximal linear approximation error δ^{lin} which is evaluated in Appendix A to

$$\delta^{\text{lin}} = a/2(\Delta x/2)^2. \tag{A.1}$$

This reflects the error due to the finite number of neurons. Let us now consider the other source of the approximation error, the finite information in the weights, and thresholds, i.e., the error due to the finite resolutions of the system variables.

3.2. The Resolution Error

To calculate the resolution error due to the number of bits with (1.3) for w_i, t_i, W_i, and T, we first have to define the range V_w, V_t, V_W, and V_T of the variables (see Brause, 1991). The maximal resolution error δ_s of a variable s in one state is just half of the resolution increment d_s in eqn (1.3)

$$\delta_s = d_s/2 = \frac{1}{2} V_s 2^{-I_s} \tag{3.8}$$

where I_s denotes the number of bits (the information) associated with the variable s. In the present approximation function example, our information distribution system parameters $c_1, \ldots, c_k$ are represented by the number of bits per variable I_w, I_t, I_W, and I_T and the number m of neurons in the first layer. In Appendix B, the error δ^{res} due to the finite resolutions I_w, I_t, I_W, I_T, and m is evaluated to

$$\delta^{\text{res}} = 2aX_1\Delta x[\delta_w X_1 + \delta] + m\delta_W + \delta_T. \tag{B.2}$$

3.3. The Optimal Information Distribution

As we have already mentioned, we are not interested in minimizing the mean squared error. Besides, since we do not assume anything about the input probability distribution $p(x)$, we cannot compute the mean squared error.

Instead, as a performance measure of the approximation network, let us compute the maximal absolute error which can occur. The maximal approximation error δ_f is given by the worst case condition that the maximal linear approximation error δ^{lin} and the maximal resolution error δ^{res} do not compensate each other but add up to

$$\delta_f = \delta^{\text{lin}} + \delta^{\text{res}}. \tag{3.9}$$

The whole information I_{sys} contained in the network is the sum of the information $m(I_w + I_t)$ of the m weights and thresholds in the first layer and the information $mI_W + I_T$ of the m weights and the threshold in the second layer

$$I_{\text{sys}} = m(I_w + I_t + I_W) + I_T. \tag{3.10}$$

When we add some information to the system by augmenting the number m of neurons, the resulting approximation will be better and, naturally, the approximation error will diminish. When we add some neurons but reduce the information in the weights and thresholds, such as to conserve the overall system information, the result is not so clear. In Figure 6 the approximation error is shown on a logarithmic scale for different values of m and constant system information I_{sys} = 708.45 bits; the number of bits for all other variables are the same $I_w = I_t = I_W = I_T$ and can be directly computed by eqn (3.10).

The minimal error of $\delta_f = 2.28 \times 10^{-3}$ is at $m^* = 16.2$ neurons and $I_T = 14.2$ bits, about 3% worse than

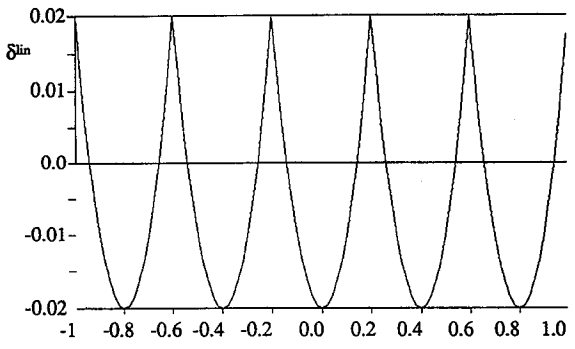


FIGURE 5. The approximation error δ^{lin} in the interval $x \in [-1, +1]$ for m = 5 neurons. As seen, the error is maximal in the middle and at the borders of the intervals.

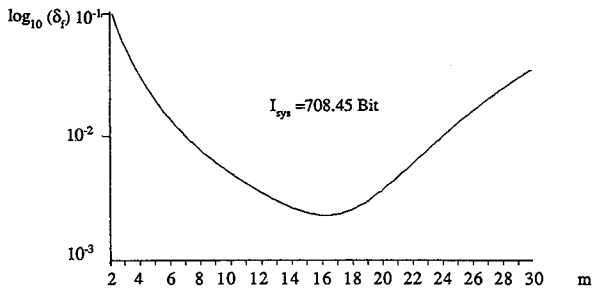


FIGURE 6. The maximal approximation error δ_f at constant system information I_{sys}. The error is drawn on a logarithmic scale.

with the optimal system parameters (see the following example). To get the optimal parameters, we just have to compute the conditions for the multivariate minimum of $\delta_f(m, I_w, I_t, I_W, I_T)$ which have already been solved in Sections 2.1 and 2.2. The condition (2.2) for an optimal information distribution becomes

$$\frac{\partial}{\partial m}(\delta^{\text{lin}} + \delta^{\text{res}})\left(\frac{\partial I_{\text{sys}}}{\partial m}\right)^{-1} = \ldots$$

$$= \frac{\partial}{\partial I_T}(\delta^{\text{lin}} + \delta^{\text{res}})\left(\frac{\partial I_{\text{sys}}}{\partial I_T}\right)^{-1} \quad (3.11)$$

with the derivatives of (3.10)

$$\frac{\partial I_{\text{sys}}}{\partial m} = I_w + I_t + I_W$$

$$\frac{\partial I_{\text{sys}}}{\partial I_w} = m = \frac{\partial I_{\text{sys}}}{\partial I_t} = \frac{\partial I_{\text{sys}}}{\partial I_W} \quad (3.12)$$

$$\frac{\partial I_{\text{sys}}}{\partial I_T} = 1.$$

The five terms of eqn (3.11) are all equal, giving us four equations with five variables. In Brause (1991), this is evaluated giving us the three equations

$$I_t = I_w + C \quad \text{with} \quad C := \log_2((X_1 - X_0)/X_1) \quad (3.13)$$

$$I_W = I_t + C \quad (3.14)$$

$$I_T = I_W + \log_2(g_T(m)/2) - \log_2((X_1 - X_0)^2/m) \quad (3.15)$$

and the equation for the number of neurons

$$m = h(m, I_T)^{1/3}. \quad (3.16)$$

This we can use for numerically given I_T as an iteration formula at the $(s + 1)$-th iteration for m:

$$m(s + 1) = h(m(s), I_T)^{1/3}. \quad (3.17)$$

Since the derivative of $h(m)^{1/3}$ is lower than 1, the convergence condition is satisfied and the iteration converges.

Example: Let us choose an information of 16 bit in the threshold T. Therefore, in the simple case of $X_0 = -1$, $X_1 = +1$, $a = 1$, $b = 0$, we have with $I_T = 16$ bit the optimal configuration at $m = 16.54$ neurons, $I_W = 14.95$ bit, $I_t = I_W - C = 13.95$ bit and $I_w = I_t - C = 12.95$ bit. The overall information in the network with eqn (3.10) is then $I_{\text{sys}} = 708.45$ bits (the same I_{sys} as in the example above) and the overall approximation error is $\delta_f = 2.213 \times 10^{-3}$ with the pure linear approximation error part of $\delta^{\text{lin}} = 1.83 \times 10^{-3}$. If we augment the information capacity of the system and use $I_T = 32$ bit, the overall error will diminish to $\delta_f = 1.847 \times 10^{-6}$ when we use the optimal system parameters.

The example of the approximation of a simple quadratic function is quite instructive to evaluate but has the disadvantage that it is not very common in real world applications. The question is, whether the proposed principle of information distribution works in a more realistic environment.

3.4. The Approximation of Robot Manipulator Control

For this purpose, let us consider the more complicated task of robot manipulator position control. The *kinematic* control computes the Cartesian position $\mathbf{x}$ of the endpoint of a robot manipulator, composed of several segments and joints by a straightforward matrix multiplication (*homogeneous transformation*) between all segment-matrices when the joint coordinates (joint angles) θ_i are given. The inverse transformation, the *inverse kinematics,* does the inverse task: When the absolute Cartesian coordinates $\mathbf{x}$ of the endpoint (e.g., the palm of the robot hand) is given, it computes the appropriate joint coordinate θ_i for each segment.

The inverse kinematic is a quite complicated function and not easy to find. When the rotational axes of the joints are oriented not in parallel or orthogonal, it is very hard or quite impossible to find an analytical solution. This fact prohibits the exploration of user-defined robot architectures and limits the adaptation of robot architectures to the user's needs.

A very promising approach is to *learn* the nonlinear mapping of inverse kinematic. One of the existing approaches by neural network systems is the use of Kohonen's (1984) Topology-Conserving Maps by Ritter, Martinetz, and Schulten (1989). Since the mapping is very coarse for a small amount of neurons, they additionally use a linear approximation with learned coefficients. In Figure 7, a neural network for the robot control is shown which models the main steps of the algorithm. The whole workspace $\{\mathbf{x}\}$ of the robot is divided into m segments, each one defined by a $\mathbf{w}^k$, the center of the workspace segment k. Instead of one exclusively activated neuron c in the Kohonen net, which has the smallest distance $|\mathbf{x} - \mathbf{w}^c|$, there is a cluster of three neurons which becomes active when their segment of the workspace is concerned. This cluster has to generate the three-dimensional difference vector $\mathbf{x} - \mathbf{w}^c$ for the linear approximation in the workspace segment by the second layer. By the learning rules, the constant input unit of each cluster (black dot in Figure 7) gen-

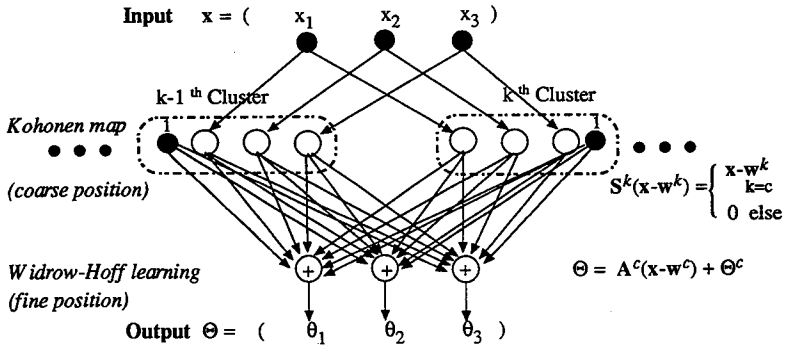


FIGURE 7. A two-layer approximator network for the inverse kinematic control of robots. Each cluster responds exclusively only in one workspace segment (*winner-take-all* network) with one set of learned joint angles corresponding to the center of the workspace segment. Additionally, the second layer interpolates linear this nonlinear mapping by the learned matrix $\mathbf{A}^k$.

erates a constant term $\Theta^c = (\theta_1^c, \theta_2^c, \theta_3^c)$, which is the vector of the angles corresponding to $\mathbf{w}^c$.

Thus, we have a two-layer approximation network again. Since the performance of this approach depends heavily on the resolution of the neural net and the resolution of the internal representation, we have to apply our methods of Section 2 to prevent an exhaustive need for storage. Instead of a one-dimensional problem with m neurons, the number of neurons grow by $m = n^3$ having n neurons per dimension. Here we have to balance the number n of storage cells (number of neurons) against the bits per cell (resolutions I_w, I_θ, I_A of the weights and coefficients). The choice for the system parameters n, I_w, I_θ, I_A can be done by the information distribution principle introduced above most efficiently.

For this purpose, let us assume that the stochastic approximation process of the Kohonen mapping has become stable and the mapping has perfectly converged. Nevertheless, there remains an error due to the discrete approximation of the nonlinear function. For the example of the commonly used PUMA robot (Figure 8) this was evaluated in Brause (1989) based on the strategy for optimal storage distribution. The main results are given below.

Let us first evaluate the maximal error δ^{lin} due to the linear approximation. Since we have only rotational axes in the system, one of the most difficult and important tasks for the manipulator is a linear, straight movement as it is often required in applications. Therefore, we consider the error on a straight line through the whole cubic work space of the manipulator. This resembles a cut through the error-weighted workspace. The numerically computed approximation error is shown in Figure 9(a). The parameter of the approx-

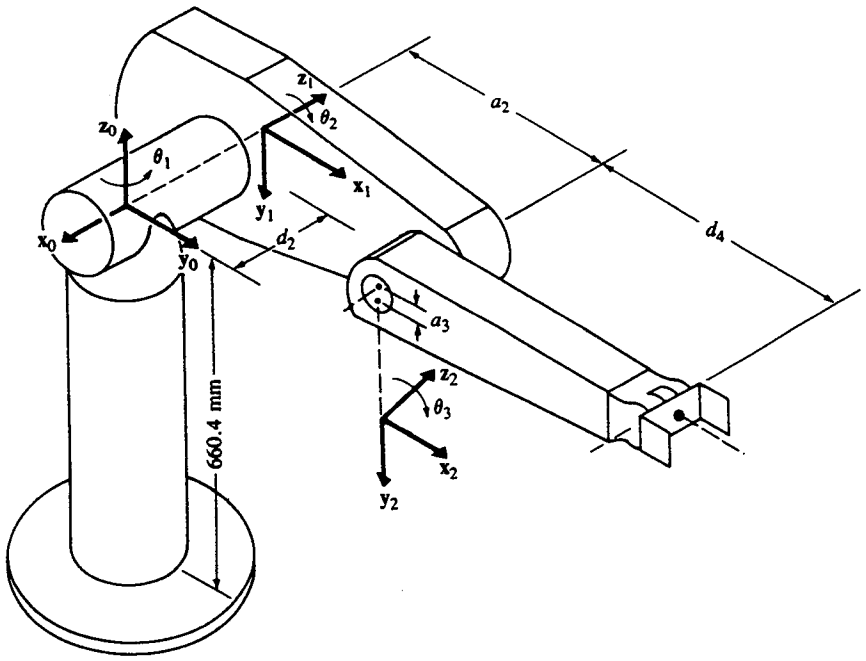


FIGURE 8. The PUMA robot (after Fu, Gonzales, & Lee, 1987). The palm of the robot hand which is subject to the inverse kinematic is marked by a dot.

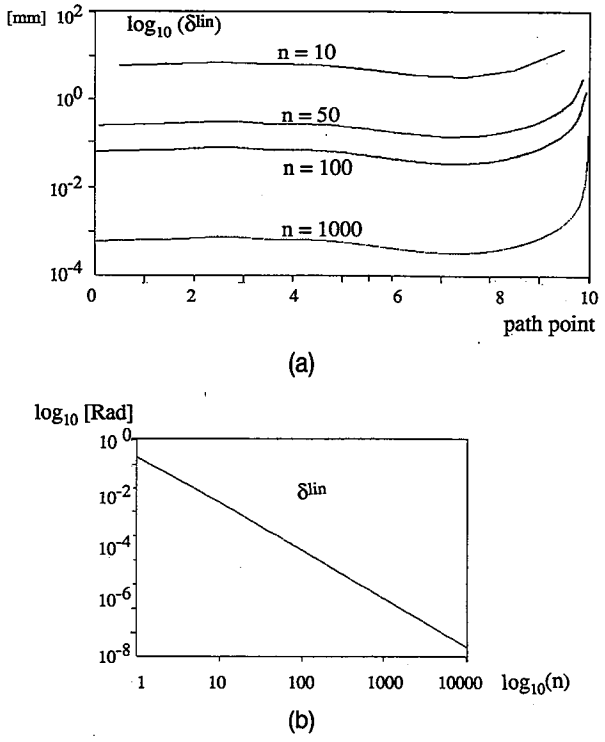


FIGURE 9. (a) The absolute Cartesian positioning error due to the linear approximation on a linear path through work space, shown on a logarithmic scale, (b) the joint error due to the linear approximation as a function of the number of neurons per dimension. Both axes are logarithmically scaled.

imation error is n, the number of neurons in one dimension. Since the robot works in three dimensions, we have $m = n^3$ neurons in the whole system.

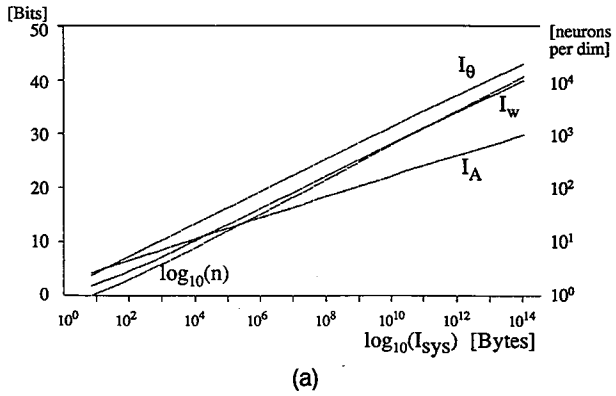
Interestingly, the lines of the different parameter values $n = 10$, 100, 1000 seem to be shifted vertically with the same offset. A numerical evaluation of the error on the positioning point with the maximal error (approximately at the third path point) shows us that this is right; in Figure 9(b) on the right hand side, the logarithm of the joint error is drawn versus the number n of the neurons in logarithmic scale. This gives us the analytical expression of $\delta^{lin} = cn^b$ as a good approximation with numerical obtained values for c and b. This coincidences well with the analytical expression (A.2) for the linear approximation error of the example of a quadratic function. The resolution error δ^{res} of the linear approximation scheme can be easily calculated by the same ideas as for eqn (B.2).

The maximal error is, again, the superposition of the error of the linear approximation and the resolution error

$$\delta_f = |\underline{\delta}^{lin} + \underline{\delta}^{res}|$$

with $\underline{\delta}^{lin}$ and $\underline{\delta}^{res}$ denoting the error vectors. Since the form of both errors are now analytically known, the conditions for the optimal information distribution of eqn (2.2) can be calculated, using the derivatives of δ_f, i.e., of δ^{lin} and of δ^{res}. Of the resulting three conditions for four parameters, all can analytically be solved except the condition for m, which was numerically iteratively approximated. The optimal system parameter values for a fixed amount of system information are shown in Figure 10(a) on the left hand side.

Now we have an optimal configuration of all system parameters yielding the minimal possible information storage amount for a given Cartesian error. The Cartesian error as a function of this optimal storage is shown in Figure 10(b) on the right hand side for the situation when all weights and thresholds are forced to have the same resolution (number of bits per variable) but optimal n and, furthermore, when they all have different, optimal resolutions.



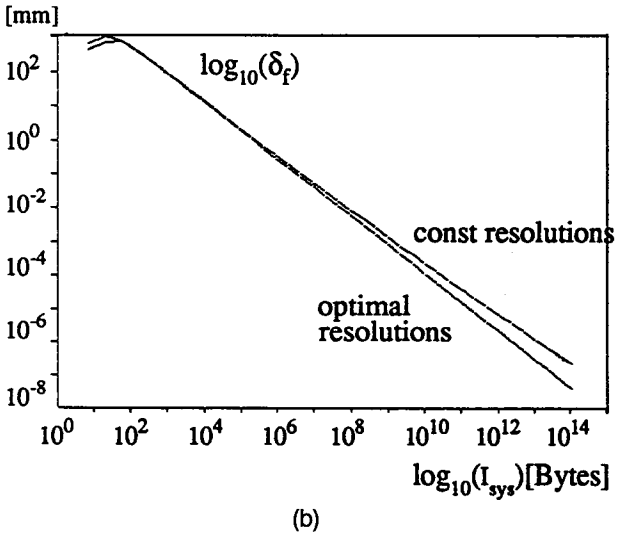


FIGURE 10. (a) The optimal values for the network parameters when the available system information is given; and (b) the resulting Cartesian error at an optimal information distribution. For the same amount of information, the approximation error of two possible configurations are drawn in a logarithmic scale: One with all weights having the same resolution and one with different, optimally computed resolutions. The optimal number m of neurons results as a function of the other parameters. For huge storage sizes, the difference between the two approaches is remarkable, but for moderate, more realistic requirements the difference can be neglected.

For a reasonable error of 0.2 mm, a value which is in the range of normal mechanical inaccuracy of the PUMA manipulator, the necessary 1.9 MB of storage memory is contained in $m = 39.6^3$ neurons and a constant resolution of $I_w = 16.4$ bits for all weights. The optimal configuration with different weight resolutions gives only a 18% smaller error, and therefore do not encourage the use of multiplication operations with variable accuracy which would be necessary in this case.

It should be noted that Figure 10 assumes real-valued number of bits and neurons. Certainly, in real applications we must use integer values (truncated or rounded) for all parameters which will result in a slightly different optimum and increased approximation error. The best selection will choose of the possible integer tupels (I_w, m^*) the one with the smallest error δ_f.

Experiments with a computer-simulated neural network controlling a real PUMA-like robot confirm the considerations above.

4. CONCLUSION

The error-bounded descriptional complexity of approximator networks is determined by the principle of optimal information distribution. This is a criterion for the efficient use of the different information storage resources in a given network. Furthermore, it can be used as a tool to balance the system parameters and to obtain the optimal network parameter configuration according to the minimal usable storage amount for a maximal error which is given.

In this paper, two examples are presented. First, a simple nonlinear function approximation is evaluated, the conditions for optimal system configuration are stated, their solutions are analytically computed, and their nature is explained. Second, the more complicated function of the inverse kinematic of a PUMA robot is considered and the results for optimal system parameters, which are partially obtained by numerical iterative approximations, are shown.

The benefits of the proposed method are not limited to real networks but apply also to all computer simulations. Here we have a tool to tailor the storage requirements according to the application needs in an optimal way.

REFERENCES

Baker, G. & Gollub, J. (1990). *Chaotic dynamics: An introduction.* Cambridge, MA: Cambridge University Press.

Brause, R. (1989). Performance and storage requirements of Topology-Conserving Maps for robot manipulator control. Internal Report 5/89, Fachbereich Informatik, University of Frankfurt, Federal Republic of Germany.

Brause, R. (1991). Approximator networks and the principle of optimal information distribution. Internal Report 1/91, Fachbereich Informatik, University of Frankfurt, Federal Republic of Germany; and *Proceedings of ICANN-91*, Amsterdam, Elsevier, North-Holland.

Gallager, R. (1968). *Information theory and reliable communication.* New York: John Wiley & Sons.

Girosi, F., & Poggio, T. (1990). Networks and the best approximation property. *Biological Cybernetics,* **63,** 169–176.

Hornik, K., Stinchcomb, M., & White, H. (1989). Multilayer feedforward networks are universal approximators, *Neural Networks,* **2,** 359–366.

Fu, K. S., Gonzales, R. C., & Lee, C. S. (1987). *Robotics.* New York: McGraw Hill.

Kohonen, T. (1984). *Self-organisation and associative memory.* Berlin: Springer-Verlag.

Kolmogorov, A. N., & Tihomirov, V. M. (1959). ε-Entropy and ε-Capacity of sets in functional spaces. *Uspehi Mat (N.S.),* **14,** 3–86. In *American Mathematical Society Translations* (1961), Series 2, **17,** 277–364.

Lapedes, A., & Farber, R. (1987). Nonlinear signal processing using neural networks: Prediction and system modelling. Los Alamos: Los Alamos National Lab., report LA-UR-87-2662.

Li, M., & Vitányi, P. (1990). Kolmogorov complexity and its applications. In J. van Leeuwen (Ed.), *Handbook of theoretical computer science.* Amsterdam: Elsevier, North Holland; Vol. A:89.

Ritter, H., Martinetz, T., & Schulten, K. (1989). Topology-Conserving Maps for learning visuomotor-coordination. *Neural Networks,* **2/3,** 159–167.

Shannon, C. E., & Weaver, W. (1949). *The mathematical theory of information.* Urbana, IL: University of Illinois Press.

Williamson, R. (1991). ε-Entropy and the complexity of feedforward neural networks. In R. Lippmann, J. Moody, & D. Touretzky (Eds.), *Advances in neural information processing systems* 3. San Mateo, CA: Morgan Kaufmann.

APPENDIX A: THE LINEAR APPROXIMATION ERROR

The nonlinear function in the interval $[x - \Delta x/2, x + \Delta x/2]$ is

$$f(x) = ax^2 + b$$

and the linear approximation by the neural network is

$$\hat{f}(x) = \alpha x + \beta \quad \text{with} \quad \alpha := 2ax.$$

The approximation error δ_1 is (see Figures 3 and A.1)

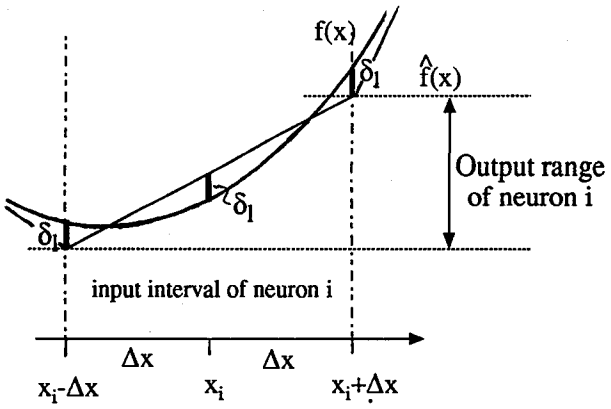


FIGURE A.1. The error of the linear approximation of a quadratic function. Remark that the errors occur in the middle and at the borders of the interval.

$$\delta_1(x) = f(x) - \hat{f}(x) = ax^2 + b - 2axx - \beta = b - \beta - ax^2 =: d$$

$$\delta_1(x_i + \Delta x/2) = f(x_i + \Delta x/2) - \hat{f}(x_i + \Delta x/2) = d + a(\Delta x/2)^2$$

$$\delta_1(x_i - \Delta x/2) = f(x_i - \Delta x/2) - \hat{f}(x_i - \Delta x/2) = d + a(\Delta x/2)^2.$$

The errors at the borders are equal. The maximal error $\max(|\delta_1(x_i)|, |\delta_1(x_i + \Delta x/2)|)$ is minimal when the errors are equal

$$|\delta_1(x_i)| = |\delta_1(x_i + \Delta x/2)| \text{ or } |d| = |d + a(\Delta x/2)^2|. \quad \text{(A.0)}$$

This is given when

$$d = -a/2(\Delta x/2)^2.$$

Therefore, the *maximal linear error* δ^{lin} depends not on the value of x_i, it is the same in the whole interval

$$\delta^{\text{lin}} = a/2(\Delta x/2)^2. \quad \text{(A.1)}$$

Since we have $\Delta x = (X_1 - X_0)/m$ we get

$$\delta^{\text{lin}} = m^{-2}a(X_1 - X_0)^2/8 = cm^b$$

$$\text{with} \quad c = a(X_1 - X_0)^2/8) \quad \text{and} \quad b = -2. \quad \text{(A.2)}$$

Note: For this approximation problem, the maximal approximation error is minimized when we divide the whole interval $[X_0, X_1]$ into m equal segments. This can easily be proven by the following: Let us regard the interval segment which has the maximal approximation error. According to eqns (A.0), the maximal linear error depends not on the value of x_i, the middle point of the i-th interval segment, but only on the length Δx_i of the segment. Thus, all the segments can be sorted into a descending order of both their length and their associated error.

Now, if we reduce the segment length Δx_i and increase the length Δx_k of the next segment in the order, the maximal error diminishes until it becomes equal to the error of the next segment. Then both segment lengths and errors are equal. A further reduction of Δx_i alone will not change the maximal error; we have to reduce both the segment lengths Δx_i and Δx_k, and have to increase the length of the third segment in the order until all three errors and segment lengths become equal.

Let us assume that this is true for the n first segments in the initial order. Then the idea above is also valid for the $n + 1$-th reduction step to the $n + 1$-th segment: By complete induction all segments have to be equal for the minimum of the maximal error, given in eqn (A.1).

APPENDIX B: THE RESOLUTION ERROR

For the computation of the resolution error let us assume that in all weights and thresholds the maximal increment error δ has occurred. The resolution and therefore the maximal increment error in one variable should be independent of its index. Then the approximating function becomes with eqns (3.2) and (3.3)

$$\hat{f}(x, \delta) = \sum_i (W_i + \delta_W)S(z_i + \delta_z) + T + \delta_T$$

$$= \sum_i W_i S(z_i + \delta_z) + T + \sum_i \delta_W S(z_i + \delta_z) + \delta_T. \quad \text{(B.1)}$$

Because the intervals are exclusive, for the k-th interval we have to regard only the influence of one neuron of the first layer; for $i < k$ we have $S(z_i) = S(z_i + \delta_z) = 1$ and for $i > k$ we have $S(z_i + \delta_z) = 0$.

$$\hat{f}(x, \delta) = \left(\sum_i^{k-1} W_i\right) + W_k S(z_k + \delta_z) + T$$

$$+ \left(\sum_i^{k-1} \delta_W\right) + \delta_W S(z_k + \delta_z) + \delta_T$$

$$= \hat{f}(x) + W_k\delta_z + (k-1)\delta_W + \delta_W S(z_k + \delta_z) + \delta_T.$$

The *maximal error* δ^{res} is encountered at $\max(x) = X_1$ on the boarder of the interval $[X_0, X_1]$. The contribution of the term $\delta_W S(\cdot)$ becomes maximal δ_W when $S\cdot) = 1$. Therefore, we have

$$\hat{f}(X_1, \delta) = \hat{f}(X_1) + (m-1)\delta_W + W_m\delta z_m + \delta_W S(z_m + \delta_z) + \delta_T$$

$$= \hat{f}(X_1) + m\delta_W + W_m\delta_z + \delta_T$$

and so with $\delta_z = \delta_w x_m + \delta_t$ we get

$$\delta^{\text{res}} = \hat{f}(X_1, \delta) - \hat{f}(X_1) = m\delta_W + W_m(\delta_w X_1 + \delta_t) + \delta_T.$$

With eqn (3.6c) we get

$$\delta^{\text{res}} = 2aX_1\Delta x[\delta_w X_1 + \delta_t] + m\delta_W + \delta_T. \quad \text{(B.2)}$$

NOMENCLATURE

$\mathbf{x}$	input data vector to the net
$\hat{f}(\mathbf{x})$	actual output of the net
$f(\mathbf{x})$	desired output
δ_f	the maximal approximation error in a compact interval
δ^{lin}	the maximal error due to a linear approximation
δ^{res}	the maximal error due to the finite number of bits per weight
V_s	value range of parameter s
I_s	resolution (number of bits) of parameter s
I_{sys}	total information (number of bits) to represent the net
$K_{\hat{f}}(f\|\mathbf{x})$	descriptional complexity
$K_{\hat{f},\epsilon}(f\|\mathbf{x})$	error-bounded descriptional complexity
m	number of neurons in the net
n	number of neurons per dimension
w_i	weight i of the first layer
t_i	threshold i of the first layer
W_j	weight j of the second layer
T	threshold of the second layer neuron
y_i	output of neuron i of first layer
z_i	activation of neuron i of first layer
$S(\cdot)$	output function
c_i	general network parameter
$L(\cdot)$	function of the approximation error with Lagrange multipliers